REMARKS

The undersigned notes with appreciation that claims 2-7 and 9-17 have been identified as being drawn to allowable subject matter.

Claims 1 and 16 have been amended to address the rejection made under 35 U.S.C. 112, second paragraph. Specifically, claim 1 has been amended at line 2 to address an issue related to proper antecedent basis, and has been amended at line 5 to more accurately refer to a previously recited component. In addition, claim 16 has been amended at line 2 to address an issue related to proper antecedent basis.

The preamble of claim 1 has been amended to better reflect the described subject matter. The scope of the claim is essentially unchanged.

Claims 1 and 8 have been rejected as being anticipated by U.S. Patent 5,515,524 to Lynch. This rejection is traversed.

U.S. Patent 5,515,524 to Lynch is representative of the state of the art prior to the present invention as discussed on pages 1-4 of the application. All methods for validating and generating configurations are methods to obtain solutions of the same problem. Facile analysis based on the problem statement or comparing assertion of a solution are not sufficient. The methods under consideration here necessarily employ the same gross steps contained in the problem definition so distinction must be based on how each of those steps individually and in aggregate are accomplished.

The methods of McDermott, <"R1: A Rule-Based Configurer of Computer Systems", John McDermott, *Artificial Intelligence* 19, (1982), pp 39-88>, and Lynch are based on improvement of a common underlying framework for navigating the set of possible configurations. The method of our invention relies on fundamentally different assumptions. A fundamental element of the prior art (and the cited reference of Lynch) is that knowledge of the aggregate system of components being built is necessary for generation and validation of configuration. For example, R1 rules of the form "If contains component X then must contain components Y,Z", or a definition in Lynch that "a computer contains CPU,Memory,Storage, ...". The representational form of these rules, constraints, or hierarchical definitions is a multi-place predicate, i.e. an expression that depends on more than one available component. In rule based systems these expressions are typically bound, i.e. the components represented by X, Y, Z are explicitly named in the rule. Constraint based systems generically associate components with classes such the constraints contain unbound variables to which any member of the class can be bound and the constraint evaluated. Lynch extends the constraint notion by distinguishing structural and functional constraints, and allowing for hierarchical classification of components and sub-systems.

Due to this fundamental element:

Some valid configurations are not recognized: it implicitly constrains the configurations generatable and recognizable as valid to a subset consistent with the implicit system level knowledge.

Some invalid configurations may be generated as valid: it may generate and validate configurations that are not valid because of the reliance on the implicit system level knowledge .

Generation and validation takes place in the context of a hierarchy of predefined system types (products or product families). Valid configurations that require changing the system type are not discovered.

The problems of maintenance associated with R1, and pointed out by Lynch, are <u>mitigated but not removed</u> by Lynch. This problem is associated with dependencies between the multi-place predicates and cannot be removed in principle. It can be argued that by introducing multiple hierarchies, Lynch increases the dependency and the effort involved in discovering it in order to maintain the database. The correlation between the structural and functional hierarchies leads to the same problems at the same model scale (size), although constraint/hierarchy based models may represent a much larger configuration space than similarly sized rule based models.

With regard to Claim 1: compared to Lynch, this invention separates the information for generating and validating configurations from questions of overall system definition or "applicability to purpose" by clearly separating all aggregate system of components level information from the component information. Claim 1 requires selecting a set of components that have a single base component having only sink interfaces, and defining an interface for each component where the component is characterized as having a source or a sink interface and properties. The only information contained in the configuration system are single place predicates describing the properties of a single component. This is a distinction with any of the referenced methods in addition to that described in Lynch.

A configuration using a method based on single place predicates is clearly distinguishable from one based on multi place predicates. The "base", atomic, unitary, smallest, non-decomposable components, are defined. All configurations are constructs of these base components. The base components may contain predicates describing functional, nonfunctional, or appropriate use characteristics of the components. These are used only in determining applicability to purpose, and are not used for validation of the configuration. All information about aggregate systems of components is strictly constructive, i.e. no new information is added to aggregates beyond the details of their aggregation. In the language of the invention, the aggregate system of components is characterized by its available source and sink interfaces (see claim 1). As with the base components, an aggregate is specifically not characterized by any functional type or classification.

The introduction and summary of the present application describe how the single place predicates in the form of source and sink interfaces generate and validate configurations. Every possible valid configuration can be discovered and no invalid configurations will be generated. This is the case even when the number of possible configurations is arbitrarily large ("too large to practically enumerate"). As explained on page 4 of the application at lines 23-26, “the problem solved by this invention is validation of a specific system or device configuration when the set of all valid configurations is too large to validate individually, enumerate in practice, or when the set is variable over time, and therefore cannot be known in advance”.

In the present invention, generation and validation does not take place in the context of any predefined hierarchy. Each base component is independent of every other base component and can be inserted or removed from the data base of available components without any impact on any other information in the data base. Predefined aggregation components will require update if one of their base components has been changed. This provides a mechanism that serves the same purpose as the sub-systems discussed in Lynch, <u>but allows for them to be</u>

regenerated as the set of base components change. Because the searchable configuration space is larger using the method of this invention, more alternatives can be presented to the user during generation. For example, if the user wished to add a third disk drive in a box having only two physical drive interfaces the generation would recognize the oversubscribed interface and be able to suggest adding an external housing or replacing the box component. This appears to require restarting with a different product family in Lynch. Thus, with the present invention, the problems with maintenance are removed. Because there are no interdependencies in the information in the component data base, base components can be added or removed at will, and with no impact on the generation and validation of new systems. As described in the application specification, a further advantage is that because no system level knowledge is maintained in the component data base, update configurations of old systems can be generated and validated by retaining only component data. Because interface information is provided by the component manufacturer, it is very likely that configurations updating or incorporating competitor or "one of" products can be generated and validated without having to first create the appropriate hierarchies with the present invention.

With regard to Claim 8, the interfaces described by Lynch, as inferred from their teachings, are specific to cables connecting the interfaces, while the interfaces of Claim 8 are for every physical mounting, logical interaction, protocol interaction, electrical connection, thermal connection, spatial relation or other possible and relevant relationship between every component. In this regard, the "connections", i.e. the cables, of Lynch are simply another component in this invention. Discovering required cables was an important function of R1, showing that the simple existence of matching physical interfaces is prior art to Lynch. The "matching property" (see claim 1, "establishing connections between components having source and sink interfaces with matching properties", which Lynch does not explicitly teach, is a new and unobvious differentiating characteristic which distinguishes the claimed invention from Lynch.

In view of the above, claims 1-17 should be in condition for allowance. Reconsideration and allowance of the claims at an early date is requested.

Respectfully submitted,

[illegible]

Michael E Whitham
Reg. No. 32,635

Whitham, Curtis & Christofferson, P.C.
11491 Sunset Hills Road, Suite 340
Reston, VA 20190

Tel. (703) 787-9400
Fax. (703) 787-7557